A parallel and matrix free framework for global stability analysis of compressible flows

O. Henze, M. Lemke, J. Sesterhenn

Institut für Strömungsmechanik und Technische Akustik, Technische Universität Berlin

An numerical iterative framework for global modal stability analysis of compressible flows using a parallel environment is presented. The framework uses a matrix-free implementation to allow computations of large scale problems. Various methods are tested with regard to convergence acceleration of the framework. The methods consist of a spectral Cayley transformation used to select desired Eigenvalues from a large spectrum, an improved linear solver and a parallel block-Jacobi preconditioning scheme.

stability analysis · compressible flow · matrix free methods

1 Introduction

In fluid dynamics, it is crucial for many technical applications to obtain stability information of the flow. While the stability of certain flows like the inviscid hyperbolic-tangent velocity profile is well studied [Blu70, Mic64], the global stability analysis of general flows is still a daunting task. Traditionally a stability analysis by e.g. ARPACK [LSY98] is performed by obtaining a linearized matrix representation of the discretized operator and using it to perform the analysis.

This kind of computation is usually limited by the available memory and computational resources. But certain physical phenomena require a highly resolved discretization and high order discretization schemes, for example the computation of acoustic modes generated by a compressible flow. This leads to an immense memory requirement during a global stability analysis, since it requires information about the describing operator in the whole computational domain. This encourages the use of a matrix free method to perform the stability analysis, which lessens the required memory significantly.

This article presents a framework which allows a matrix free computation of a global linear temporal stability analysis of compressible flows using a high order spatial discretization of the compressible Navier-Stokes equations. Compared to previous applications of this method, for example by [Mac09], the numerical stability analysis can be performed in a parallel manner.

We focus on the the temporal linear stability analysis of compressible flows. Being able to compute the stability analysis without the need of an operator in matrix formulation leads to a considerable reduction in required memory during a computation. In turn, certain challenges which arise because of the matrix free method have to be considered.

The presented framework employs iterative Krylov methods in order to obtain the desired Eigeninformation. Krylov methods are a natural choice for this kind of problem since they can be parallelized well and can operate without explicit knowledge of the operator. The base method of choice is the implicitly restarted Arnoldi method (IRAM) [Sor02]. It is combined with a spectral Cayley transformation to facilitate in the selection of the correct Eigenvalues and to provide a speedup of the computation [MSR94]. The use of this transform introduces a new linear system which has to be solved. An improvement of a common iterative method (GMRES) is tested as well as a preconditioning scheme in order to improve convergence.

Validation of the method is performed on a compressible mixing layer setup using a tangent hyperbolic profile as investigated by Blumen [Blu70]. The growth rate of the most unstable mode of this profile is computed using the proposed framework and compared to literature.

This article is structured as follows: Section 2 of this paper will discuss the physical model used in the stability analysis. Section 3 will explain in detail the concepts used in the stability analysis framework and the parallelization of it. To close, a model problem and results based on it are presented and evaluated in sections 4 and 5.

2 Governing equations

For the problems considered herein, the compressible Navier-Stokes (1) equations are used as the governing equations. They are given in a characteristic type formulation [Ses00] and are used in a matrix free fashion. Note that the use of this special formulation is not required for the framework, other formulations can be applied as well. The formulation introduces the primitive variables p as pressure, u_i as the velocity vector and s as entropy. Additionally, τ_{ij} describes the stress tensor, T the temperature, Φ the dissipation and ρ the density, with μ and μ_v describing the viscosity and bulk viscosity of the fluid, respectively.

$$\begin{aligned}
\frac{\partial p}{\partial t} + \varrho c^2 \frac{\partial u_i}{\partial x_i} &= \frac{p}{c_v}\left(\frac{\partial s}{\partial t} + u_i \frac{\partial s}{\partial x_i}\right) \\
\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} &= -\frac{1}{\varrho}\frac{\partial p}{\partial x_i} + \frac{1}{\varrho}\frac{\partial \tau_{ij}}{\partial x_j} \\
\frac{\partial s}{\partial t} + u_j \frac{\partial s}{\partial x_j} &= -\frac{1}{\varrho T}\left(\frac{\partial}{\partial x_j}\left(\lambda \frac{\partial T}{\partial x_j}\right) + \Phi\right) \\
\Phi &= \tau_{ij} s_{ij} \\
\tau_{ij} &= 2\mu\left(s_{ij} - \frac{1}{3} s_{kk}\delta_{ij}\right) + \mu_v s_{kk}\delta_{ij} \\
s_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)
\end{aligned} \tag{1}$$

This set of equations is closed with Sutherland's law and the ideal gas equation and uses the summation convention with $i = 1,2,3$.

Going forward, the short hand operator $\mathcal{N}(q)$ will be used to refer to the Navier Stokes equations, with q containing all primitive flow variables $q = [p, u_i, s]$.

Within the used scheme, the wave propagation terms are discretized with a sixth order finite difference scheme using compact finite difference schemes by [Lel92] on a staggered grid, which allows an accurate computation required to extract the stability information. The dissipation terms were discretized using a similar scheme of fifth order.

3 Numerical stability analysis

Ansatz and overview

In order to perform a global stability analysis, we need to formulate an appropriate ansatz which allows the computation of the temporal stability properties of the flow.

$$\dot{q} = J(q_0)q \tag{2}$$

Therein $J(q_0)$ describes the temporal evolution of the operator $\mathcal{N}_{\text{lin}} = J(q_0)$, linearized around a base state q_o. To this end we investigate the temporal stability. The goal of the stability analysis is to obtain the Eigenvalues λ of the Operator. For this, an ansatz fitting to the problem has to be chosen. The form of the ansatz depends on the structure of the investigated flow, especially on its degree of homogeneity.

For a general three dimensional flow without any degree of spatial homogeneity, the general approach (3) can be used for the stability analysis, as given by [Mac09].

$$\tilde{q}(x,y,z,t) = \phi(x,y,z)\exp(-i\alpha ct) \tag{3}$$

Herein, α denotes the wave number of the temporal disturbance, c represents the wave propagation speed, while $\phi(x,y,z)$ describes the Eigenvector corresponding to α. Combining the model equation (2) with the general approach (3) results in an Eigenvalue problem (4) which has to be solved.

$$J(q_0)\phi(x,y) = \lambda\phi(x,y) \tag{4}$$

An Eigenvalue problem of this type can be solved directly if the Jacobian matrix $J(q_0)$ is known. Since the memory requirements for storing this matrix are too high for global stability analysis problems, we have to employ matrix free methods to solve the it.

Matrix free framework

In order to compute the Eigenvalues needed for the stability analysis, the Jacobian of the operator solving the defining equations (1) is required. Since the defining equations are nonlinear and not available in a matrix formulation, an approximation of the Jacobian has to be computed. Therefore a Frechèt-type derivative (5) is employed for the linearization of the nonlinear operator during the stability analysis.

$$J(q_0)q = \frac{\mathcal{N}(q_0 + \epsilon q) - \mathcal{N}(q_0)}{\epsilon} \approx \mathcal{N}_{Lin}q \tag{5}$$

The correct choice of the parameter ϵ required for this derivative is critical to its accuracy. We used the findings on the influence of ϵ given by [SSS09] and set it accordingly to $\epsilon = \|q\|\sqrt{\epsilon_{\mathrm{m}}}$, with ϵ_{m} representing the machine accuracy of the system. The use of this linearization requires the evaluation of the defining equations $\mathcal{N}(q)$ every time the Jacobian is required during the computation. This application is usually one of the most numerically expensive operations and one should therefore try to minimize the number of required calls.

The use of a matrix free method to obtain information about J without explicitly storing it leads naturally to the use of iterative Krylov subspace methods, which do not require the Jacobian in explicit form but rather just matrix vector products of the type $J(q_0)\phi(x,y)$. These matrix vector products can be computed in a matrix free fashion by using the previously described Frechèt derivative (5).

Krylov methods

Since the framework is designed to work in a matrix free environment, a method which can operate in such an environment is required for the stability analysis. Krylov methods suite this requirement well and will therefore be used as method of choice. The idea of Krylov methods is to iteratively create a sequence of subspaces $\mathcal{K}_m$ which meets (6).

$$\mathcal{K}_1(\mathcal{N}_{\mathrm{lin}},r_0) \subset \mathcal{K}_2(\mathcal{N}_{\mathrm{lin}},r_0) \subset ... \subset \mathcal{K}_m(\mathcal{N}_{\mathrm{lin}},r_0) \tag{6}$$

As stated before, $\mathcal{N}_{\mathrm{lin}} = J(q_o)$ is a (linearized) matrix representation of the Jacobian of $\mathcal{N}(q)$. Note that due to the iterative nature of Krylov methods only the product given by the Frechèt derivative (5) is required.

The basis V_m of these subspaces can then be used to compute the Arnoldi relation (7), which projects a portion of $\mathcal{N}_{\mathrm{lin}}$ onto the Krylov subspace. This results in a Hessenberg matrix H_m that is usually much smaller than $\mathcal{N}_{\mathrm{lin}}$ and contains an approximation of the eigenvalues and eigenvectors of the operator.

$$\mathcal{N}_{\mathrm{lin}}V_m = V_m H_m + f_m e_m^H \tag{7}$$

In (7), $f_m e_m^H$ represents the last component of the residual vector of the size m Arnoldi decomposition, which is visualized in Fig. 1.

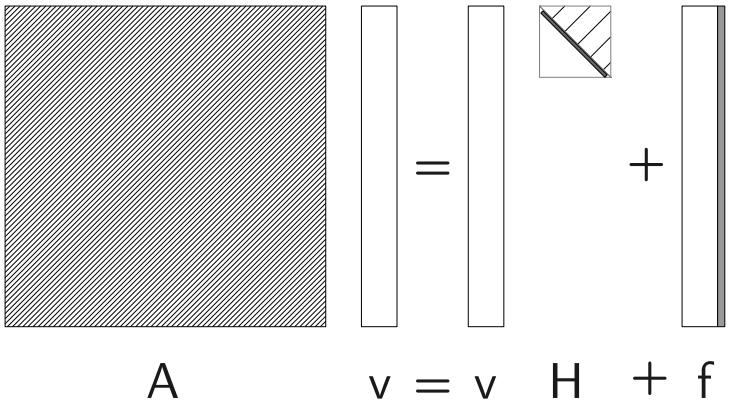

Figure 1: Visualization of the Arnoldi relation, see [Sor02].

Implicitly restarted Arnoldi method

In order to solve the eigenvalue problem (4) iteratively, the implicitly restarted Arnoldi method (IRAM) developed by [Sor02] is used. It offers a robust method for obtaining Ritz pairs - approximations to the Eigenvalues and Eigenvectors of the operator $\mathcal{N}$. Its main benefits are twofold. On one hand, it is a Krylov-type method and therefore does not need the explicit Jacobian of the operator. On the other hand, the IRAM allows a certain degree of control concerning which part of the spectrum is to be computed. This feature is especially important in order to obtain the desired results in a manageable time frame. In essence, it is a restarted Arnoldi method which allows the removal of unwanted Eigenvalue information during the restart of the iteration by using a QR-shift. A schematic overview of the algorithm is presented in algorithm 1. An important aspect of the IRAM with respect to the stability analysis is to define a suitable convergence criterion to determine when the desired Ritz pairs have been computed. As pointed out by [LS96], choosing an appropriate criterion is difficult for cases involving non-Hermitian operators which occur often when analyzing compressible flows since the residual norm $\|f_m\|$ of the Arnoldi decomposition does not necessarily imply convergence as opposed to the Hermitian case. To ensure a small backward error, a Ritz estimate (8) proposed by [LS96] is used. This criterion is applied to each computed Ritz pair $(\hat{x}, \theta)$.

$$||f_m||\,|e_m^T \hat{x}| \leq \max(\epsilon_{\mathrm{m}}||H_m||, tol_{\mathrm{iram}}|\theta|) \tag{8}$$

Cayley transformation

While the selective properties of the IRAM are very useful for this task, they are sometimes not sufficient when considering stability problems of compressible flows, which lead to problems due to clustering of Eigenvalues close to the origin or when dealing with badly conditioned flows. In order to enhance the selection properties of the framework even further, a Cayley transformation (9), as proposed by [MSR94], is applied to the Eigenvalue problem (4).

$$C(\mathcal{N}_{\mathrm{lin}}) = (\mathcal{N}_{\mathrm{lin}} - \sigma I)^{-1} (\mathcal{N}_{\mathrm{lin}} - \mu I) \tag{9}$$

This leads to a generalized Eigenvalue problem of the type

$$(\mathcal{N}_{\mathrm{lin}} - \sigma I) v_{j+1} = (\mathcal{N}_{\mathrm{lin}} - \mu I) v_j, \tag{10}$$

which has to be solved. The solution of this linear system requires an iterative solver. Due to the nature of the iterative solvers, the Cayley transformation is therefore only applied inexactly.

Algorithm 1 Implicitly restarted Arnoldi factorization [Sor02]. Parallel parts are represented by a "parallel" comment.

Start with a $m = k + p$ step Arnoldi factorization (algorithm 7)
$\mathcal{N}_{\text{lin}} V_m = V_m H_m + f_m e_m^*$ ▷ parallel
while not converged **do**
  compute spectrum(H_m), select p shifts $\mu_1, \mu_2, \ldots, \mu_p$
  perform implicit QR-Shifts:
  $Q = I_m$
  for $j = 1, \ldots, p$ **do**
    call qr factorization($H_m - \mu_j I$) $\rightarrow [Q_j, R_j]$
    $H_m \leftarrow Q_j^* H_m Q_j$
    $Q \leftarrow Q Q_j$
  end for
  update the Arnoldi factorization:
  $\hat{\beta}_k = H_m(k+1,k)$
  $\sigma_k = Q(m,k)$
  $f_k = v_{k+1}\hat{\beta}_k + f_m \sigma_k$
  $V_k \leftarrow V_m Q(:, 1:k)$
  $H_k \leftarrow H_m Q(1:k, 1:k)$
  convergence check using (8) ▷ parallel
  use obtained k-step factorization as starting point:
  $\mathcal{N}_{\text{lin}} V_k = V_k H_k + f_k e_k^*$
  apply p steps of Arnoldi method to obtain new factorization
  $\mathcal{N}_{\text{lin}} V_m = V_m H_m + f_m e_m^*$ ▷ parallel
end while

To illustrate how the Cayley transformation affects the spectrum of the transformed operator, Fig. 2 shows a comparison of a sample spectrum with and without a Cayley transformation. Depending on the parameters μ and σ, the transformation has two effects on the shape of the spectrum. It introduces a rotation as well as a stretching effect. Therefore, it can be used to separate clusters of Eigenvalues and to rotate desired Eigenvalues into the part of the spectrum with positive Eigenvalues, which are favored by Arnoldi based methods. Another important feature of this transform is to map inner Eigenvalues to the outer part of the spectrum. This is important since Arnoldi based method usually converge first onto Eigenvalues distant from the origin.

Iterative solvers

The basic solver for the solution of (10) is a typical restarted GMRES based on [SS86], which was chosen for its robust numerical behaviour. Other methods are also applicable, for example BiCGstab.

Since the solution of (10) usually incurs a large numerical effort, it is beneficial to reduce the iterations required to solve it as much as possible. We investigate the possible speedup of a modified GMRES version and compared them to the standard formulation. The analyzed LGMRES uses error approximations to augment the Krylov subspace between restart cycles [BJM05]. The variant aims at improving the speed of convergence of the traditional restarted GMRES by lessening the impact of the necessary restart.

Parallelization

In order to compute large scale stability analysis, parallelization of the framework is required. Since the framework is based on IRAM, it lends itself well to parallelization since the numerically most expensive steps, the construction of the new Krylov subspaces, can be distributed among available CPUs such that each process does not require information about the entire computational domain. This reduces the required communication between the processes significantly. The steps that require parallelization are

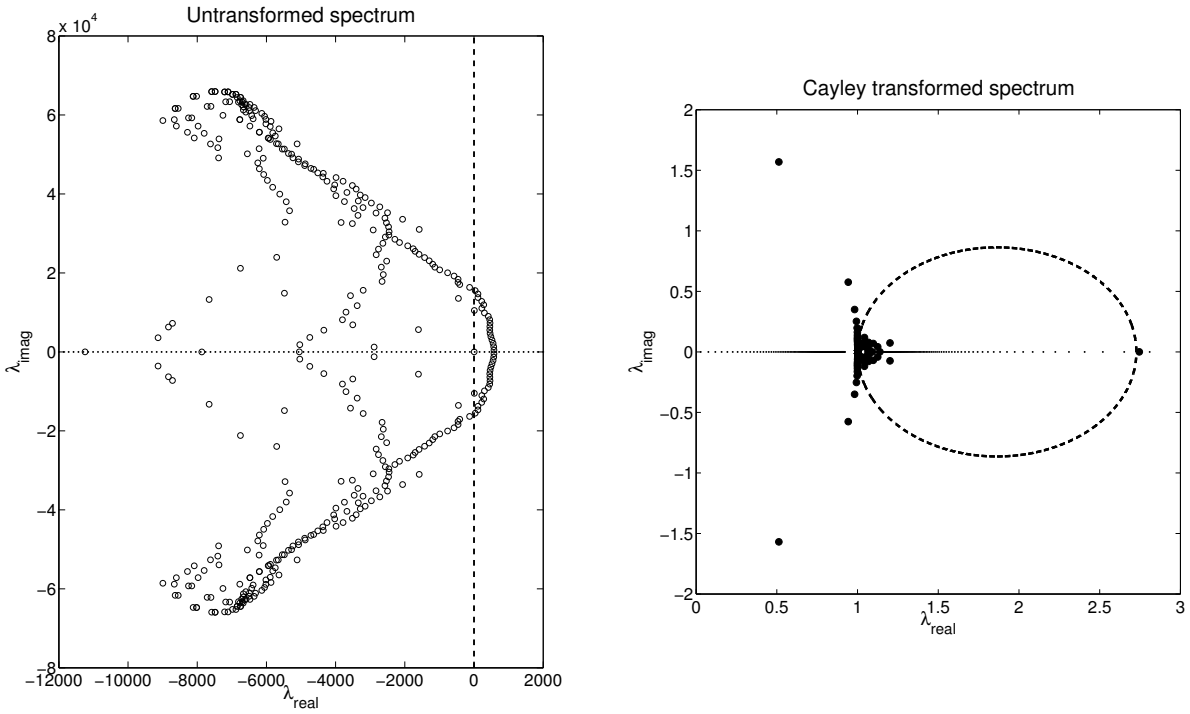

Figure 2: Comparison between an untransformed spectrum and its Cayley transformed counterpart, using $\mu = 440.5$, $\sigma = 1200.23$. Also showing transformation of two Cartesian lines.

marked in the respective algorithm scheme such as Listing 1. The implementation used in the scope of this work achieves parallelization by parallelizing the derivatives using MPI, but other means of parallelization are applicable as well.

Preconditioning

Preconditioning techniques are often necessary to improve the speed of convergence when solving large linear systems arising during the stability analysis. While many methods for preconditioning are well researched, their application to a matrix free and parallel case is not straightforward. Since a global preconditioner is difficult to assemble due to the lack of the operator matrix, a local approach for the preconditioner was chosen for this work to serve as a proof of concept for matrix free preconditioning in parallel stability analysis.

The framework uses a local block-Jacobi scheme similar to [HS91] which assembles a local preconditioner matrix on each process and uses it to accelerate the solution of the linear system. While this approach is rather simple, it serves as a starting point for future refinement.

Numerical framework overview

The proposed framework can be roughly divided into two parts, an outer iteration loop consisting of the IRAM-algorithm [Sor02], which obtains the stability information, and an inner iteration loop for the solution of the linear system required for the application of the inexact Cayley transformation. The relationship between these is depicted in Fig. 3.

An application of the operator $\mathcal{N}_{\text{lin}}$ is required for each step of the inner iteration. Each of these calls of the operator translates to one evaluation of the defining equations (1). This is usually the step which requires the most computational resources. It is also the "innermost" step of the framework and therefore the most called.

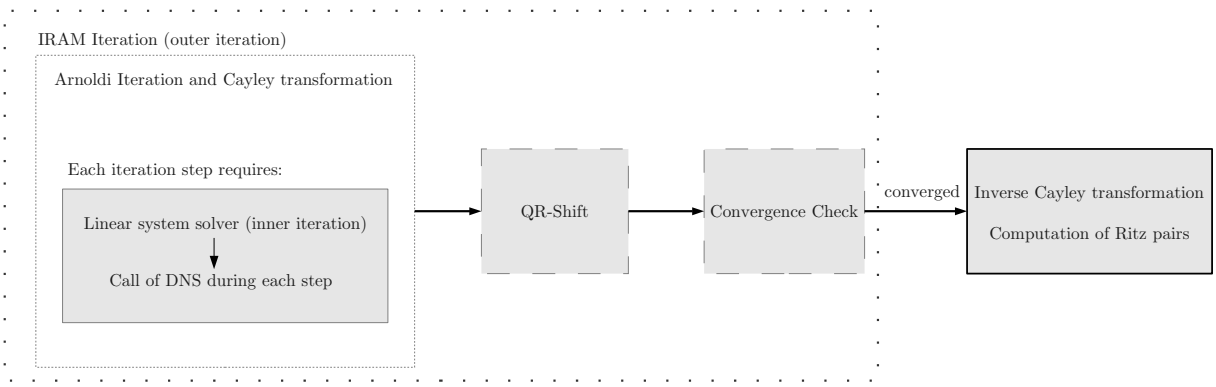

Figure 3: Schematic overview of the stability framework.

4 Model problem

To illustrate the proposed framework, a model problem is used in the scope of this paper. We investigate the temporal stability of two-dimensional compressible mixing layer using a hyperbolic tangent velocity profile similar to the works of Blumen [Blu70]. The profile used is described by equation (11), which imposes a velocity profile in the x_1-component of the velocity onto the stream. The thickness of the shear layer is controlled by the parameter δ.

$$u_1(x_2) = u_\infty \tanh\left(\frac{x_2 - 0.5 \cdot L_2}{\delta}\right), \tag{11}$$

with u_∞ as reference velocity and L_2 as domain length in x_2 direction. The model problem is set up in a computational domain with 128×512 points as depicted in Fig. 4. It employs a periodic boundary condition on the left and right boundaries as well as a non-reflective boundary condition on the lower and upper boundaries. Additionally, grid stretching was used to improve the resolution in the critical parts of the domain where the mixing effect occurs. This leads to a minimum cell distance in x_2 - direction of $\approx 0.08\delta$ centered towards to highest velocity gradients present in the profile.

The tangent hyperbolic velocity profile is seeded into the domain as a starting condition. In order to test the parallel aspect of the framework, the computation is distributed between at least 4 processors.

This model problem is investigated with regard to its linear temporal stability. The goal is to identify the least stable mode by using the previously outlined framework. The domain length in x_1-direction L_1 should fit at least one entire wavelength of the desired mode, and was chosen to be exactly one wavelength of the least stable mode scaled by mixing layer thickness δ. Assuming a wavenumber α of the least stable mode, the length of the domain in x_1-direction would result as $L_1 \frac{2\pi\delta}{\alpha}$. The domain length in y-direction was chosen as 8 times L_1.

Compared to the traditional matrix based approach on stability analysis, the presented matrix free framework requires significantly less memory. Consider the above model problem with a 128×512 points resolution. Assuming double precision, storing the operator alone would require $(128 \cdot 512)^2 \cdot 64\text{Bit} \approx 32\text{GB}$ of memory. When performing a parallel computation, this requirement is shared between all available nodes. The memory requirements of the matrix-free approach are tested in the following Sec. 6.

5 Validation

The proposed framework is validated using the model problem described in the previous section. The least stable modes for a various settings are computed and their growth rates compared to literature,

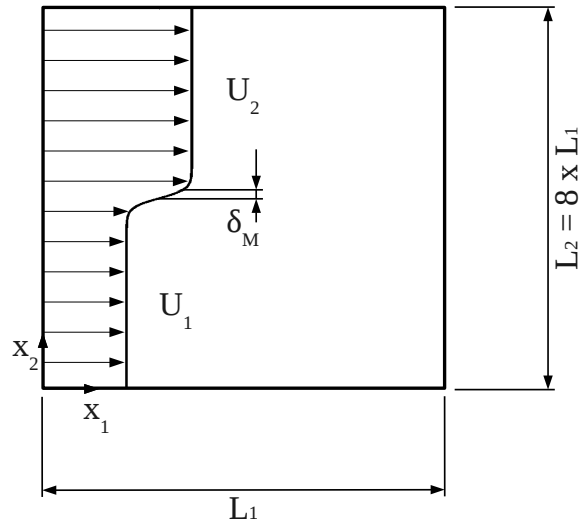

Figure 4: Draft of the used computational domain showing dimensions and the hyperbolic tangent profile.

see [Blu70, Mac09]. All computations are performed in parallel and cover the different types of iterative solvers described in the previous section as well as the preconditioner introduced previously.

Results - Validation

The proposed framework is able to compute growth rates of the least stable mode in good accordance to the results given by Blumen [Blu70] and Mack [Mac09]. The detailed results are compared in Tbl. 2. The different configurations are designed to cover a range of Mach numbers as well as different resolutions at a high Reynolds number. The different configuration parameters are listed in Tbl. 1. In addition to the computed growth rates, an exemplary mode computed during the validation is presented in Fig. 5.

From this results we find the growth to be computed in good accordance with literature.

Configuration	M	Re	Resolution
V01	0.1	$\to \infty$	64×256
V02	0.5	$\to \infty$	64×256
V03	0.9	$\to \infty$	64×256
V04	0.5	1000	64×256
V05	0.1	$\to \infty$	16×64
V06	0.5	$\to \infty$	16×64
V07	0.9	$\to \infty$	16×64
V08	0.5	1000	16×64

Table 1: Configuration parameters for different validation setups.

6 Numerical Experiments

As mentioned previously, the computational costs and required memory of the stability problem increases rapidly when considering problems which require high resolutions. While the memory requirements are mitigated to a great extent by the use of a matrix-free framework, the computational effort required is still significant. It is therefore important to reduce this computational effort, for example by improving the convergence behavior of the stability framework to avoid unnecessary iterations.

Configuration	computed growth rate ω	ω by Blumen	ω by Mack
V01	0.189	0.187	0.187521
V02	0.1411	0.141	0.1411
V03	0.0547	0.055	0.054723
V04	0.1273	n/a	0.1271
V05	0.189	0.187	0.187521
V06	0.139	0.141	0.141167
V07	0.0540	0.055	0.054723
V08	0.1270	n/a	0.1271

Table 2: Results of the validation for various setups.

This section therefore investigates the effectiveness of the preconditioning method and iterative solvers with regard to speed up the iteration based on the results of numerical experiments. Since one of the main objectives of the presented framework is its matrix-free and parallel implementation, we will discuss the scalability and especially memory aspects based on the numerical experiments.

Lastly, while the choice of Cayley parameters has an influence on the performance of the framework, the presented numerical experiments do not cover a study on these parameters, since the focus of this work is to provide an overview of the framework. For a more detailed description of the effects of the Cayley transform, the reader is referred to [MSR94] or [Mac09]. Still, a few general remarks on the choice of Cayley parameters for this case are summarized at the end of this section.

The numerical experiments are based on the model problem discussed previously and differ in resolution, type of preconditioning and iterative solver used to cover a variety of possible combinations.

All further experiments share the following basic parameters: A Mach number M = 0.5, Reynolds number Re $=\rightarrow \infty$, domain length in x_1- direction $L_1 = 15.83\delta$, tolerance of the IRAM $tol_{\text{iram}=1e-5}$ and tolerance for the GMRES-type method $tol_{\text{gmres}} = 1e-6$. The subspace size m of the IRAM was varied from $100 - 250$, depending on the resolution of the case, with lower resolutions using smaller subspace sizes.

Preconditioning Scheme

Aside from the growth rate as a measure of the correctness of the computation, the number of required matrix-vector product operations is recorded. Since these correlate directly with the amount of required operator evaluations - the by far most numerically expensive part in these simulations. They serve as a good measure to compare different methods regarding speedup. Fig. 6 compares different computations, using the Block Jacobi preconditioner as well as running without preconditioning. A reduction in needed iterations for the preconditioned cases is clearly visible in the comparison. The reduction differs from case to case, but usually ranges 40-60%.

Aside from the speedup aspect another equally important aspect of the preconditioning became apparent during the numerical experiments. Since the stability problem arising from compressible flows is usually conditioned very poorly, computations without preconditioning often fail to produce accurate results. The use of a preconditioner alleviated this problem by enabling the computation of viable results in these cases and making the computation more robust. The resolutions used for the results above was therefore chosen small to allow a comparison to the unpreconditioned case.

Linear Solvers

Aside from the speedup achieved by the preconditioning, the speedup of the LGMRES solver is also investigated by numerical experiments. Table 3 has exemplary results for the effectiveness of the LGRMES-method compared to standard GMRES.

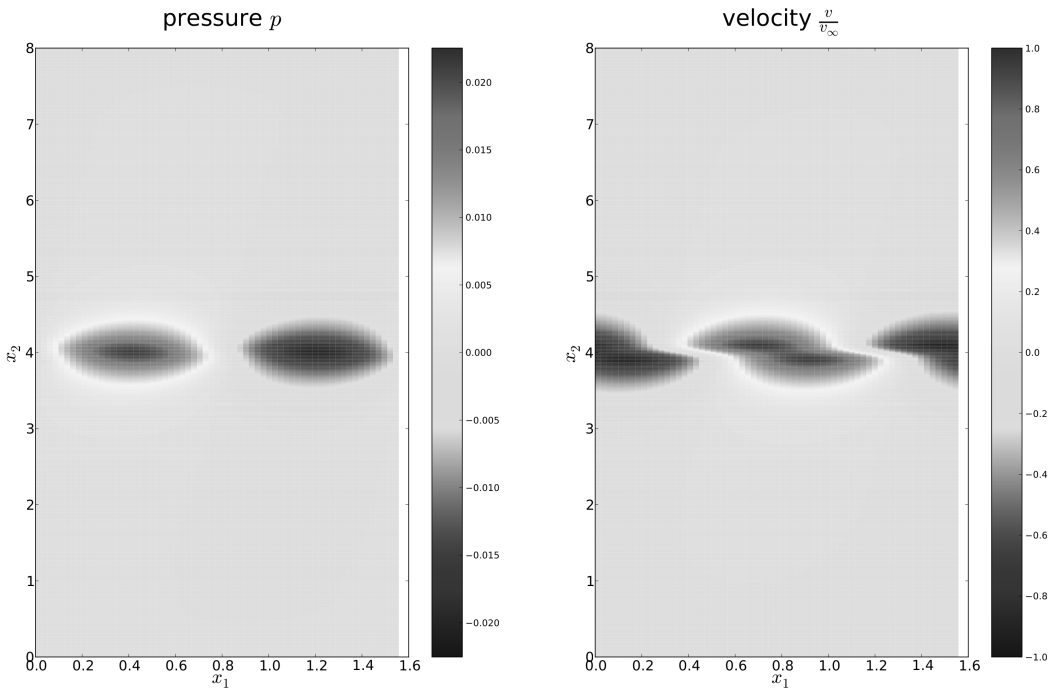

Figure 5: Least stable mode computed with resolution 128×512 using 64 CPUs. One period in x-direction is shown. Depicted are fields pressure p and the velocity component $u_2 = v$. Also, the figure shows the underlying grid used in the computation.

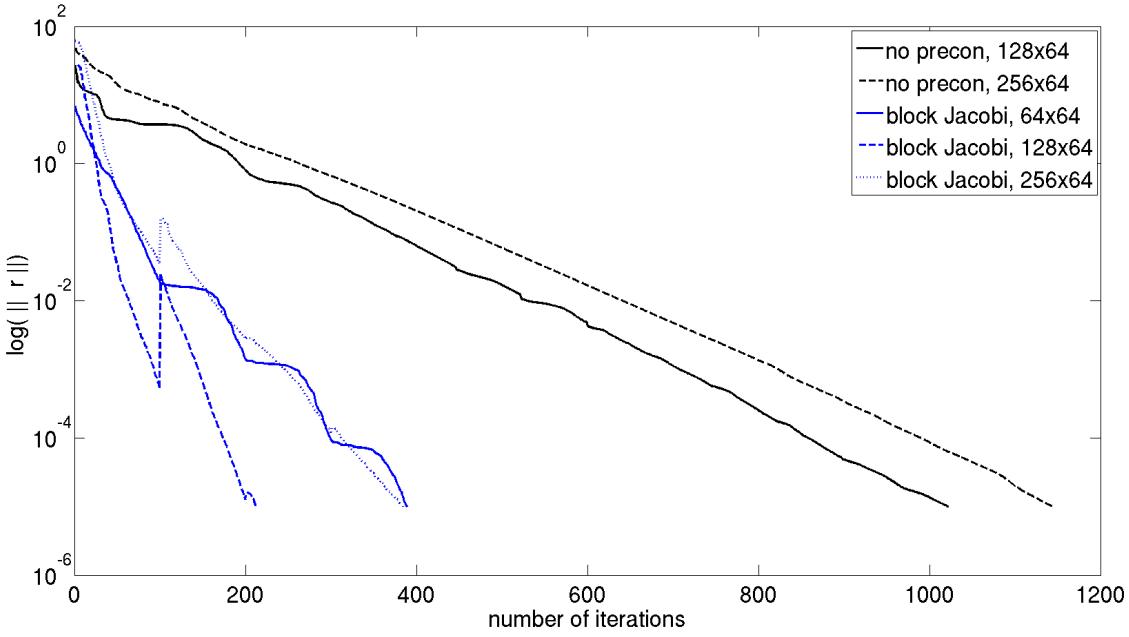

Figure 6: Comparison of the iteration process of one call of the linear solver for five different setups using LGMRES.

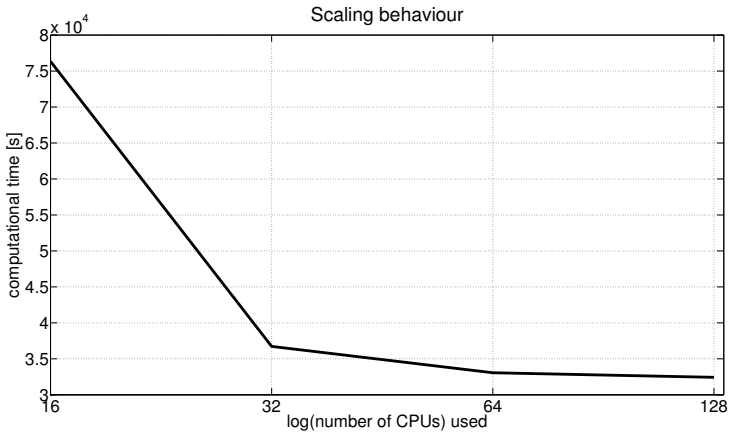

Figure 7: Scaling behavior of the framework.

In comparison to the effectiveness of the preconditioner, the speed up achieved by this method is rather small. Given the very low additional numerical effort incurred by this method compared to standard GMRES, the use of LGMRES is still advisable.

Memory and computation time considerations

In order to highlight the effects of the different methods with respect to speedup, a comparison between selected setups is displayed in Tbl. 3. The table compares the different configurations in terms of required iterations as well as required memory during the computation. Note that the computation using the higher resolution was not possible without the use of a preconditioner. The memory requirement displayed therein is estimated analytically. The table contains several important aspects of the framework. The first to be

	resolution	GMRES	memory*	LGMRES	memory*
no precond.	256×64	1419325	0,1GB*	1389364	0,1GB*
no precond.	1024×128	/		/	
block Jacobi	256×64	65164	4.1GB	63472	4.1GB
block Jacobi	1024×128	1459950	256,4GB	1411357	256,4GB

Table 3: Iteration times for different solvers, preconditioning methods and resolutions for the model problem. (*) The memory requirement is shared across the nodes.

noted is the memory requirement of the method is heavily influenced by the method of preconditioning. In its current iteration, the use of the Block Jacobi method requires a significant amount of memory when investigating higher resolutions compared to the unpreconditioned case. This is a trade-off for the drastic reduction in required iterations compared to the unpreconditioned computations. For the examples provided in the given table, the preconditioned computation required roughly 20 times less iterations, a significant speedup even when considering the memory requirements of this method. However, the memory requirement is shared over the parallelized nodes.

As an additional metric, the scaling of the computational time and the memory requirement is collected in Fig. 7 to give an example for the scaling behavior of the framework when using the Block-Jacobi preconditioner. The scaling observed in this case massively deteriorates from 64 processors onward. This can be explained by the implementation of the preconditioner which requires a huge amount of calls of $\mathcal{N}(q)$ (ergo the right hand side of the considered equation) to compose the preconditioner prior to its application. With increasing number of processors used, the numerical costs of the composition surpass the costs of the solution of the preconditioned linear system, since the calls of $\mathcal{N}(q)$ require MPI calls. While this process is parallelized, it still acts as a bottleneck when using a relatively large number of processors

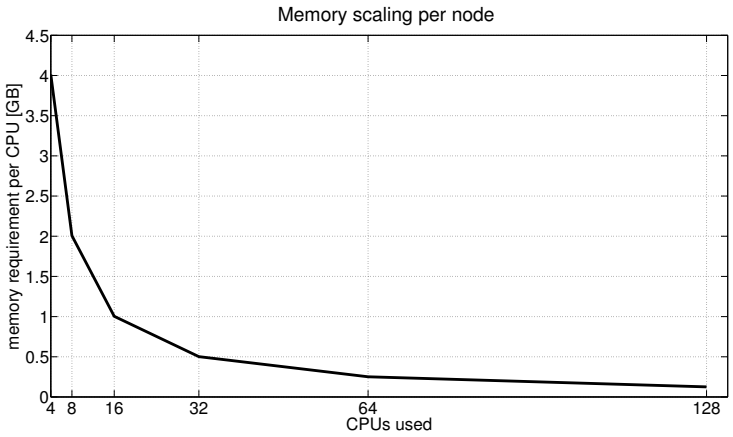

Figure 8: Theoretical memory scaling of the framework with preconditioning.

- for the parameters given, this number is at 64 processors. Furthermore, the preconditioner cannot be omitted despite deteriorating the scaling of the framework since the preconditioner is a requirement to enable the correct computation in the first place.

Due to this behavior, the improvement of the preconditioner seems to be the most promising route to consider for improving the performance of the framework. As with the framework in general, the goal is to reduce the number of required right hand side calls. This could be achieved by a using non compact, lower order discretization of the operator $\mathcal{N}(q)$. This would significantly reduce the MPI communication during operator calls while also having the possible benefit of reducing the memory required during the composition of the preconditioner, [Mac09]. Fig. 8 shows the estimated memory requirement per node when using preconditioning. The actual memory requirements during numerical experiments the scale nearly linearly with the number of used processors, except for a small overhead which results in a sub-linear scaling. This is likely dependant on the method of parallelization used in the framework, although this was not investigated in the scope of this work.

Cayley parameters and further remarks

The numerical experiments performed for this work generated further observations which are not covered in the above sections, yet can still be useful to get a better understanding of the problem at hand and to serve as guidelines for common problems.

Firstly, the condition of the linear systems deteriorates with increasing resolution. This is to be expected since an increase in resolution leads to a higher density of computable modes in the mixing layer. This results in an increased clustering of Eigenvalues which can affect the condition of the linear system. Additionally, a higher resolution can also result in an increase of unstable modes introduced by the discretization scheme. There are several ways to lessen the impact of these phenomena.

A first measure to be considered when dealing with the increased resolution is to increase the sizes of the subspaces available to the IRAM and GMRES. This step is usually required to account for the increased complexity of the computed spectrum and comes with an increase in memory and computational resources. For the resolutions investigated in this works, subspace sizes varying from 100 to 250 were found adequate.

In addition to the increase in subspace size, an adjustment of the Cayley parameters can also lead to an improvement of the computation for higher resolutions. Unfortunately, the method for determining the correct parameters is not as straightforward compared to the subspace size since the Cayley parameters are seemingly not correlating with the resolution in a simple fashion. For the resolutions considered herein, the following parameter sets in Tbl. 4 have been used successfully. While the Cayley parameters can influence the number of iterations required by the linear solver, the effect of fine tuning the parameters was not found significant.

resolution	recommended parameters
32×128	$\mu = 230.5$, $\sigma = 800.32$
32×256	$\mu = 444.5$, $\sigma = 1200.32$
64×512	$\mu = 444.5$, $\sigma = 1200.32$
128×1024	$\mu = 444.5$, $\sigma = 1200.32$

Table 4: Recommended parameters for different resolutions of the mixing layer model problem.

7 Conclusions

A framework for the matrix free and parallel stability analysis of compressible flows has been presented and validation results and speedup experiments have been showcased. To enhance the performance of the framework, a straightforward parallel preconditioning method was presented and tested successfully. The methods presented herein can be seen as a proof of concept for the matrix free stability analysis in parallel environments.

Furthermore, the numerical experiments provide several ways that can be explored for a further increase in efficiency of the computation. As mentioned previously and judging from the results of the numerical experiments, the preconditioning scheme appears to be the most promising route to explore for further improvements. Two aspects can be considered as an angle for improvements to the preconditioning scheme.

Firstly, the construction of the preconditioner itself could be handled in a more sophisticated manner regarding memory requirements and method of construction. For example, rather than using the whole local operator as base for the preconditioning matrix, one could revert to using a reduced operator which still covers the physics but uses a much lower order finite difference scheme, as employed by [Mac09] in the non parallel case. This would reduce the memory required by the preconditioner while also facilitating its construction.

Secondly, the inversion and application of the preconditioning matrix offers room for improvement, e.g. by using sparse techniques for saving the matrix if the structure of the matrix is suitable.

While two different linear solvers were compared, the impact of the chosen method was found to be small compared to the impact of the preconditioning method. Therefore, while other methods could still be investigated with respect to speed up, the study of the preconditioning method seems to be more efficient, especially.

Acknowledgements

The authors gratefully acknowledge support by the Deutsche Forschungsgemeinschaft (DFG) as part of collaborative research center SFB 1029 “substantial efficiency increase in gas turbines through direct use of coupled unsteady combustion and flow dynamics”

References

[BJM05] A. H. Baker, E. R. Jessup, and T. Manteuffel. A technique for accelerating the convergence of restarted gmres. *SIAM J. Matrix Anal. Appl.*, 26(4):962–984, April 2005.

[Blu70] W. Blumen. Shear layer instability of an inviscid compressible fluid. *Journal of Fluid Mechanics*, 40:769–781, 1970.

[HS91] M. Hegland and P. E. Saylor. Block jacobi preconditioning of the conjugate gradient method on a vector processor, 1991.

[Lel92] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16–42, 1992.

[LS96] R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications*, 17(4):789–821, 1996.

[LSY98] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6. Siam, 1998.

[Mac09] C. Mack. *Global stability of compressible flow about a swept parabolic body.* PhD thesis, Ecole Polytechnique, 2009.

[Mic64] A. Michalke. On the inviscid instability of the hyperbolictangent velocity profile. *Journal of Fluid Mechanics*, 19:543–556, 1964.

[MSR94] K. Meerbergen, A. Spence, and D. Roose. Shift-invert and cayley transforms for detection of rightmost eigenvalues of nonsymmetric matrices. *BIT Numerical Mathematics*, 34(3):409–423, 1994.

[Ses00] J. Sesterhenn. A characteristic-type formulation of the navier–stokes equations for high order upwind schemes. *Computers & fluids*, 30(1):37–67, 2000.

[Sor02] D. C. Sorensen. Numerical methods for large eigenvalue problems. *Acta Numerica*, 11:519–584, 2002.

[SS86] Y. Saad and M. H. Schultz. Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, July 1986.

[SSS09] J. Schulze, P. Schmid, and J. Sesterhenn. Exponential time integration using krylov subspaces. *International journal for numerical methods in fluids*, 60(6):591–609, 2009.